END TRANSACTION

```
END [OF] TRANSACTION [operand1 ...]
```

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
Operand1	C	S			N	A	N	P	I	F	B	D	T					yes	no

Related Statements: GET TRANSACTION DATA | BACKOUT TRANSACTION | STORE | UPDATE | DELETE

Function

The END TRANSACTION statement is used to indicate the end of a logical transaction. A logical transaction is the smallest logical unit of work (as defined by the user) which must be performed in its entirety to ensure that the information contained in the database is logically consistent.

Successful execution of an END TRANSACTION statement ensures that all updates performed during the transaction have been or will be physically applied to the database regardless of subsequent user, Natural, database or operating system interruption. Updates performed within a transaction for which the END TRANSACTION statement has not been successfully completed will be backed out automatically.

The END TRANSACTION statement also results in the release of all records placed in hold status during the transaction.

The END TRANSACTION statement can be executed based upon a logical condition.

For further information, see the section Database Access in the Natural Programming Guide.

Databases Affected

An END TRANSACTION statement *without* transaction data (that is, without *operand1*) will only be executed if a database transaction under control of Natural has taken place. Depending on the setting of the Natural profile parameter ET (see your Natural Operations documentation) the statement will be executed only for the database affected by the transaction (ET=OFF), or for all databases that have been referenced since the last execution of a BACKOUT TRANSACTION or END TRANSACTION statement (ET=ON).

An END TRANSACTION statement *with* transaction data (that is, with specifying *operand1*) will always be executed and the transaction data be stored in a database as described in the following section. It depends on the setting of the ET parameter (see above) for which other databases the END TRANSACTION statement will be executed.

Storage of Transaction Data - operand1

For a transaction applied to an Adabas database, or to a DL/I database in a batch-oriented BMP region (in IMS environments only), you may also use this statement to store transaction-related information. These transaction data must not exceed 2000 bytes. They may be read with a GET TRANSACTION DATA statement.

The transaction data are written to the database specified with the profile parameter ETDB (see your Natural Operations documentation).
If the ETDB parameter is not specified, the transaction data are written to the database specified with the profile parameter UDB - except on mainframe computers: here, they are written to the database where the Natural Security system file (FSEC) is located (if FSEC is not specified, it is considered to be identical to the Natural system file, FNAT; if Natural Security is not installed, the transaction data are written to the database where FNAT is located).

Considerations for DL/I Databases

Because PSB scheduling is terminated by a "syncpoint" request, Natural saves the PSB position before executing the END TRANSACTION statement. Before the next command execution, Natural re-schedules the PSB and tries to set the PCB position as it was before the END TRANSACTION statement. The PCB position might be shifted forward if any pointed segment had been deleted in the time period between the END TRANSACTION and the following command.

Considerations for SQL Databases

As most SQL databases close all cursors when a logical unit of work ends, an END TRANSACTION statement must not be placed within a database modification loop; instead, it has to be placed after such a loop.

Considerations for VSAM Databases

For information on the transaction logic that applies when accessing VSAM, see the Natural for VSAM documentation.

Restriction

This statement cannot be used with ENTIRE SYSTEM SERVER.

Example 1

```
/* EXAMPLE 'ETREX1S:' END TRANSACTION (STRUCTURED MODE)
/*******************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 CITY
  2 COUNTRY
END-DEFINE
/*******************************************************
FIND EMPLOY-VIEW WITH CITY = 'BOSTON'
 ASSIGN COUNTRY = 'USA'
 UPDATE
 END TRANSACTION
/*******************************************************
 AT END OF DATA
  WRITE NOTITLE *NUMBER 'RECORDS UPDATED'
 END-ENDDATA
/*******************************************************
END-FIND
END
```

```
7 RECORDS UPDATED
```

Equivalent reporting-mode example: See the program ETREX1R in the library SYSEXRM.

Example 2

```
/* EXAMPLE 'ETREX2:' END TRANSACTION (WITH ET DATA)
/************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 FIRST-NAME
  2 CITY
1 #PERS-NR (A8) INIT <' '>
END-DEFINE
/************************************************************
REPEAT
 INPUT 'ENTER PERSONNEL NUMBER TO BE UPDATED:' #PERS-NR
  IF #PERS-NR = ' '
   ESCAPE BOTTOM
  END-IF
/************************************************************
FIND EMPLOY-VIEW PERSONNEL-ID = #PERS-NR
 INPUT (AD=M)    NAME / FIRST-NAME / CITY
 UPDATE
 END TRANSACTION #PERS-NR
END-FIND
/************************************************************
END-REPEAT
END
```

```
ENTER PERSONNEL NUMBER TO BE UPDATED: 20027800
```

```
NAME LAWLER
  FIRST-NAME SUNNY
  CITY MILWAUKEE
```

